TEST-TIME REINFORCEMENT LEARNING FOR GUI GROUNDING VIA REGION CONSISTENCY

Yong Du[1,2,*], **Yuchen Yan**[1,*], **Fei Tang**[1], **Zhengxi Lu**[1], **Chang Zong**[3], **Weiming Lu**[1], **Shengpei Jiang**[4] **Yongliang Shen**[1,†]
[1]Zhejiang University
[2]Central South University
[3]Zhejiang University of Science and Technology
[4]SF Technology
diong666@csu.edu.cn {yanyuchen,syl}@zju.edu.cn

GitHub: https://github.com/zju-real/gui-rcpo
Project: https://zju-real.github.io/gui-rcpo

ABSTRACT

Graphical User Interface (GUI) grounding, the task of mapping natural language instructions to precise screen coordinates, is fundamental to autonomous GUI agents. While existing methods achieve strong performance through extensive supervised training or reinforcement learning with labeled rewards, they remain constrained by the cost and availability of pixel-level annotations. We observe that when models generate multiple predictions for the same GUI element, the spatial overlap patterns reveal implicit confidence signals that can guide more accurate localization. Leveraging this insight, we propose **GUI-RC (Region Consistency)**, a test-time scaling method that constructs spatial voting grids from multiple sampled predictions to identify consensus regions where models show highest agreement. Without any training, GUI-RC improves accuracy by 2-3% across various architectures on ScreenSpot benchmarks. We further introduce **GUI-RCPO (Region Consistency Policy Optimization)**, which transforms these consistency patterns into rewards for test-time reinforcement learning. By computing how well each prediction aligns with the collective consensus, GUI-RCPO enables models to iteratively refine their outputs on unlabeled data during inference. Extensive experiments demonstrate the generality of our approach: GUI-RC boosts Qwen2.5-VL-3B-Instruct from 80.11% to 83.57% on ScreenSpot-v2, while GUI-RCPO further improves it to 85.14% through self-supervised optimization. Our approach reveals the untapped potential of test-time scaling and test-time reinforcement learning for GUI grounding, offering a promising path toward more robust and data-efficient GUI agents.

1 INTRODUCTION

The rapid advancement of GUI (Graphical User Interface) agents is fundamentally transforming human-device interaction, enabling users to control complex interfaces in digital devices through natural language across diverse applications (Gou et al., 2024; Tang et al., 2025c; Wang et al., 2024). At the heart of these systems lies GUI grounding, the critical capability to accurately map natural language instructions to precise pixel coordinates on UI elements (Cheng et al., 2024; Tang et al., 2025b). This fundamental task determines whether an agent can successfully execute user commands, making it the cornerstone of reliable GUI automation.

[*] The first two authors have equal contributions. This work was done when the first author was an intern at Zhejiang University.
[†] Corresponding author.

Current approaches to GUI grounding have achieved impressive results through extensive train-time optimization. These methods fall into two main categories: supervised fine-tuning (SFT) with large-scale annotated datasets (Qin et al., 2025; Wu et al., 2024) and reinforcement learning with carefully designed reward functions (Lu et al., 2025; Luo et al., 2025a; Tang et al., 2025a). However, these approaches share two fundamental limitations. First, they rely exclusively on train-time scaling while leaving test-time computation underutilized, missing opportunities for performance gains through inference-time optimization. Second, they require extensive labeled data, where each sample demands precise pixel-level annotations, creating a significant bottleneck for scaling to new domains and applications (Chu et al., 2025; Wu et al., 2025b).

This raises a critical question: ***Can we leverage test-time computation to enhance GUI grounding performance without relying on any additional labeled data?*** Recent breakthroughs in large language models have demonstrated the remarkable potential of test-time scaling (Guan et al., 2025; Snell et al., 2024; Muennighoff et al., 2025). Self-consistency (Wang et al., 2023) aggregates multiple reasoning paths through majority voting, achieving substantial improvements in mathematical reasoning. Test-time reinforcement learning (TTRL) (Zuo et al., 2025) enables models to self-improve on unlabeled data by generating experiences and computing rewards during inference. These successes in language domains suggest untapped potential for applying test-time scaling to vision-language tasks like GUI grounding.

However, adapting test-time scaling to GUI grounding presents unique challenges. Unlike text-based reasoning where outputs are discrete tokens, GUI grounding operates in a continuous, high-resolution coordinate space where minor pixel deviations can lead to incorrect element selection. The visual complexity of modern interfaces, with overlapping elements, dynamic layouts, and varying resolutions, introduces significant prediction uncertainty. The key insight is to transform this uncertainty from a limitation into an opportunity: when models generate multiple predictions for the same element, the patterns of agreement and disagreement across predictions reveal valuable information about localization confidence of the model.

In this work, we present GUI-RC (GUI Region Consistency), a test-time scaling approach that aggregates spatial information across multiple model predictions to improve grounding accuracy. Our core observation is that when sampling multiple predictions from a model, certain screen regions consistently appear across different outputs, indicating higher confidence in those locations. By constructing a spatial voting mechanism that identifies consensus regions with maximum overlap, GUI-RC achieves significant performance improvements (e.g., +2.75% on OS-Atlas-Base-7B) without any additional training or labeled data.

Building upon this foundation, we introduce GUI-RCPO (GUI Region Consistency Policy Optimization), which enables test-time training through region consistency signals. Inspired by recent advances in TTRL (Zuo et al., 2025), GUI-RCPO computes rewards based on how well each prediction aligns with the consensus across multiple samples, then uses these self-generated rewards to update model parameters during inference. This label-free optimization further improves performance (e.g., +5.5% on Qwen2.5-VL-3B-Instruct), demonstrating that models can effectively self-improve on unlabeled GUI data.

Our main contributions are:

- We propose GUI-RC, a test-time scaling method for GUI grounding that leverages spatial voting across multiple predictions to improve localization accuracy without additional training or labeled data.
- We introduce GUI-RCPO, a test-time reinforcement learning method that uses region consistency as a self-supervised reward signal, enabling models to improve grounding capabilities through policy optimization on unlabeled GUI screenshots.
- We demonstrate consistent improvements across multiple benchmarks and model architectures. GUI-RC improves accuracy by 2-3% on average, while GUI-RCPO achieves further gains of 4-5% on average through label-free optimization.
- We reveal that further applying GUI-RC after GUI-RCPO yields additional performance gains, demonstrating that our methods support progressive, self-bootstrapping improvement without external supervision, and provide a complementary alternative to train-time optimization for GUI automation.

2 Related Works

2.1 GUI Grounding

GUI grounding refers to the task of locating target elements on a screen based on natural language instructions. Given a screenshot s and an instruction i, the model M is required to output the specific position of the target element that best matches the instruction. Current approaches to this task primarily fall into two paradigms. The first formulates GUI grounding as a point prediction task, where models directly output the coordinate (x, y) of the target element. Representative works include Seeclick (Cheng et al., 2024), which enhances grounding through pre-training using automated data curation, and UGround (Gou et al., 2024), which creates massive datasets to train universal visual grounding models. More sophisticated frameworks have emerged, such as the unified vision-based framework by Xu et al. (2024) that separates GUI grounding from planning and reasoning through a two-stage training pipeline, and the dual-system framework by Tang et al. (2025b) that dynamically switches between fast prediction and systematic analysis.

While point-based prediction enables seamless integration with subsequent actions, this discrete supervision signal often fails to capture the inherent spatial ambiguity in human-GUI interaction (Wu et al., 2025b). This limitation has motivated the second paradigm, where models predict bounding boxes (x^-, y^-, x^+, y^+) representing the region that best matches the instruction. OS-Atlas (Wu et al., 2024) exemplifies this approach by developing the largest open-source cross-platform GUI grounding corpus, while Qin et al. (2025) builds end-to-end GUI agents through massive datasets and multi-stage training. Recent reinforcement learning approaches like GUI-G1 (Zhou et al., 2025) further optimize region-based grounding through box size constraints in rewards and difficulty-aware scaling in the RL objective. Despite these advances, all existing methods rely heavily on train-time optimization with labeled data. Our work takes an orthogonal approach by leveraging test-time computation to improve grounding accuracy without additional training, addressing the scalability limitations that constrain current approaches.

2.2 Test-Time Scaling

Test-time strategies for LLMs have shown that, increasing computation resource during inference time can substantially enhance output accuracy and consistency without altering model weights (Wei et al., 2022; Madaan et al., 2023; Liu et al., 2025). A key strategy is self-consistency voting, where multiple sampled outputs are aggregated to select the most frequent or confident answer (Wang et al., 2023), improving mathematical reasoning and symbolic tasks (Wei et al., 2022; Madaan et al., 2023). Building on this, search and tree-based methods (Yao et al., 2023; Muennighoff et al., 2025) explore diverse reasoning in a tree structure, enabling backtracking and global evaluation of different solution strategies. More recently, test-time reinforcement learning (TTRL) (Zuo et al., 2025) introduces reward-driven optimization during inference, enabling models to iteratively refine their outputs using feedback signals like self-verification or correctness.

Extending these ideas to vision-language tasks like GUI grounding introduces new challenges due to the continuous nature of coordinate predictions and the visual ambiguity of user interfaces. Existing test-time strategies for GUI grounding rely heavily on zoom-based refinement. Wu et al. (2025a) identifies ambiguous regions via gradient-based attribution and zooms in to refine vision-language alignment. Luo et al. (2025b) extracts and aggregates predictions from multiple zoomed-in sub-regions around a focal point. In this paper, we explores a new perspective by leveraging the spatial overlap patterns across multiple predictions. Our methods introduce a spatial voting mechanism that transforms the uncertainty in continuous predictions into reliable consensus regions, enabling effective test-time scaling that works across both point-based and region-based grounding paradigms, without requiring architectural modifications or specialized preprocessing.

3 Methods

We first formalize the GUI grounding task and identify the key challenges in applying test-time scaling to this domain (Section 3.1). We then present GUI-RC, our test-time scaling approach that leverages spatial consistency across multiple predictions to improve grounding accuracy (Section 3.2). Finally, we introduce GUI-RCPO, which extends region consistency to reward signals for test-

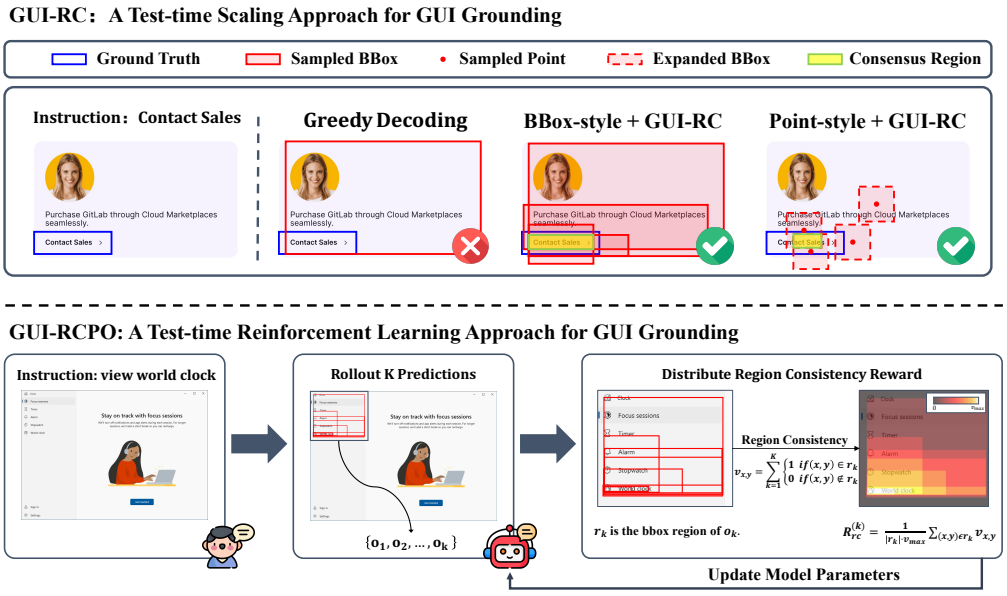


Figure 1: Overview of our test-time scaling methods for GUI grounding. **Upper**: GUI-RC aggregates K sampled predictions through spatial voting to extract a consensus region, achieving more accurate localization than greedy decoding. **Lower**: GUI-RCPO computes region consistency rewards based on the voting heatmap and uses these self-supervised signals to update model parameters, enabling label-free improvement through test-time reinforcement learning.

time reinforcement learning on unlabeled GUI data (Section 3.3). Figure 1 provides an overview of both methods.

3.1 Problem Formulation

GUI grounding aims to map natural language instructions to precise locations on graphical interfaces. Formally, given a screenshot $s \in \mathbb{R}^{H \times W \times 3}$ and an instruction i, a model M outputs the spatial location of the UI element that best matches the instruction. As discussed in Section 2.1, this task has two primary formulations:

$$\text{Point-based: } M(s, i) \to (x, y) \tag{1}$$

$$\text{Region-based: } M(s, i) \to (x^-, y^-, x^+, y^+) \tag{2}$$

Both formulations face a fundamental challenge: the continuous nature of coordinate prediction introduces inherent uncertainty, especially given the pixel-level precision required for successful interaction. This uncertainty is compounded by the complexity of modern interfaces with overlapping elements, varying resolutions, and ambiguous natural language descriptions.

Our key insight is that this uncertainty, rather than being regarded as a limitation, can be leveraged for improvement. When a model generates multiple predictions for the same input, the spatial distribution of these predictions reveals implicit confidence patterns that can guide more accurate localization.

3.2 GUI-RC: Test-Time Scaling via Spatial Voting

GUI-RC transforms the uncertainty in individual predictions into robust consensus through spatial aggregation. As illustrated in the upper part of Figure 1, while greedy decoding often fails to locate the correct element, GUI-RC successfully identifies the target by aggregating information across multiple samples. The method operates in three stages: multi-sample generation, spatial voting, and consensus extraction.

Multi-Sample Generation. Given an input pair (s, i), we sample K predictions from the model using temperature-based sampling:

$$\{M_k(s,i)\}_{k=1}^{K} \sim p_\theta(\cdot|s,i) \tag{3}$$

This sampling process naturally produces variation in predictions due to the continuous output space and model uncertainty. Crucially, we observe that despite this variation, certain screen regions appear consistently across samples, indicating higher model confidence.

Spatial Voting Mechanism. To quantify this consistency, we construct a spatial voting grid $v \in \mathbb{R}^{H\times W}$ matching the screenshot resolution. Each sampled prediction contributes votes to this grid:

$$v_{x,y} = \sum_{k=1}^{K} \mathbb{1}[(x,y) \in R_k] \tag{4}$$

where R_k represents the region from the k-th prediction. For region-based models, R_k is simply the predicted bounding box. For point-based models, we approximate the implicit attention region by expanding the predicted point (x_k, y_k) into a square region of size

$$R_k = [x_k - \alpha/2, x_k + \alpha/2] \times [y_k - \alpha/2, y_k + \alpha/2] \tag{5}$$

This expansion is necessary because point predictions, while precise, do not explicitly encode the spatial extent of the target element. As shown in Figure 1, the expanded regions for point-style models (dashed red boxes) enable meaningful spatial voting similar to bbox-style models.

Consensus Extraction. The voting grid v now encodes the collective attention of the model across samples. We extract the consensus region through a principled selection process that first identifies the maximum vote count across the entire grid as $v_{\max} = \max_{x,y} v_{x,y}$. This maximum value represents the highest level of agreement among predictions. We then find all contiguous regions where every pixel has this maximum vote count, forming the set $\mathcal{R}_{v_{\max}} = \{r : \forall(x,y) \in r, v_{x,y} = v_{\max}\}$. Among these high-confidence regions, we select the one with the largest area as our final consensus region: $\hat{r}_{\text{cons}} = \arg\max_{r\in\mathcal{R}_{v_{\max}}} |r|$. The consensus region $\hat{r}_{\text{cons}}$ represents the area where the model shows highest and most consistent attention, providing a more reliable grounding prediction than any individual sample.

3.3 GUI-RCPO: Test-Time Reinforcement Learning via Region Consistency

While GUI-RC improves performance through inference-time aggregation, we further explore whether region consistency can guide model improvement through test-time training. As illustrated in the lower part of Figure 1, GUI-RCPO transforms region consistency into a self-supervised reward signal for policy optimization.

Region Consistency as Reward. The key insight is that predictions aligning with high-consistency regions should be reinforced, while outliers should be suppressed. For each sampled prediction r_k in the rollout, we compute its region consistency reward:

$$R_{rc}^{(k)} = \frac{1}{|r_k| \cdot v_{\max}} \sum_{(x,y)\in r_k} v_{x,y} \tag{6}$$

This reward measures the average vote density within the predicted region, normalized by the region size and maximum possible votes. As visualized in Figure 1, the heatmap representation shows how different regions receive varying levels of votes, with warmer colors indicating higher consistency. Predictions that overlap with these high-vote regions receive higher rewards, encouraging the model to converge toward consensus areas.

Policy Optimization. We formulate GUI grounding as a reinforcement learning problem where the VLM acts as policy π_θ. Using Group Relative Policy Optimization (GRPO) (Shao et al., 2024), we optimize the expected region consistency reward:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(s,i)\sim D}\mathbb{E}_{r\sim\pi_\theta(\cdot|s,i)}[A(r)\log\pi_\theta(r|s,i)] \tag{7}$$

where $A(r)$ is the advantage computed from relative rewards within each group of samples. GRPO's group-relative formulation is particularly suitable for our setting as it normalizes rewards across different inputs, preventing optimization bias toward easier examples.

A unique property of GUI-RCPO is its ability to progressively improve without external supervision. As the model updates its parameters based on region consistency rewards, its predictions become more concentrated around high-confidence regions, which in turn provides stronger and more reliable reward signals for further optimization. This self-bootstrapping process continues until the model converges to a stable distribution centered on consensus regions.

4 EXPERIMENTS

4.1 EXPERIMENT SETUP

Models. We evaluate our methods on a diverse VLMs to demonstrate their generality across different architectures and training paradigms. For general-purpose models, we use Qwen2.5-VL-3B-Instruct and Qwen2.5-VL-7B-Instruct (Bai et al., 2025), as well as InternVL3-2B-Instruct and InternVL3-8B-Instruct (Zhu et al., 2025), which represent state-of-the-art vision-language models at different scales. For GUI-specific models that have been explicitly trained on GUI grounding tasks, we evaluate UGround-V1-7B (Gou et al., 2024), OS-Atlas-Base-7B (Wu et al., 2024), and UI-TARS-1.5-7B (Seed, 2025). These models span both point-based and region-based prediction paradigms, allowing us to assess the effectiveness of our methods across different output formats.

Evaluation Benchmarks and Metrics. We evaluate our methods on three GUI grounding benchmarks: ScreenSpot (Cheng et al., 2024), ScreenSpot-v2 (Wu et al., 2024), and ScreenSpot-Pro (Li et al., 2025). ScreenSpot and ScreenSpot-v2 assess model's grounding performance in general GUI environments spanning Mobile, Web, and Desktop platforms. ScreenSpot-Pro specifically focuses on high-resolution and professional interfaces. Following standard evaluation protocols, we adopt grounding accuracy as our primary metric: a prediction is considered correct if the predicted point or the center of the predicted bounding box falls within the ground-truth bounding box (Cheng et al., 2024).

Implementation Details. For GUI-RC, we sample 64 outputs using a temperature of 0.5 and a top_p of 0.95 for voting, the hyperparameter α is set to 50. For the baselines, we employ greedy decoding with temperature 0. For GUI-RCPO, we adopt the VLM-R1 (Shen et al., 2025) framework and conduct TTRL training on the Screenspot-v2 benchmark without using the ground-truth data. For each input, 16 samples are generated with a temperature of 0.7 and top_p of 0.95. We train the models for 2 epochs (approx. 40 steps) with a global batch size of 64, learning rate of 1e-6, and KL penalty $\beta = 0.04$. All training and evaluation are conducted on 8 NVIDIA A100-80GB GPUs.

4.2 MAIN RESULTS

4.2.1 EVALUATION RESULTS OF GUI-RC EXPERIMENTS

GUI-RC consistently improves the end-to-end grounding performance. We compared the performance of the base models on three benchmarks before and after applying GUI-RC. Table 1 presents the evaluation results. It can be observed that GUI-RC consistently improves the overall grounding capability across different models, regardless of its output style and whether the model is specifically trained for GUI tasks. For instance, OS-Atlas-Base-7B achieves an overall improvement of 2.75%, with a notable 6.28% increase in icon localization in mobile scenarios. Moreover, for general models like Qwen2.5-VL-3B/7B-Instruct that output in bbox-style, GUI-RC brings even greater improvements on the ScreenSpot-Pro compared to ScreenSpot and ScreenSpot-v2. This suggests that GUI-RC is particularly effective in helping models tackle more challenging grounding tasks involving high-resolution and professional GUIs.

GUI-RC achieves greater improvements when applied to bbox-style prediction models. Another observation is that GUI-RC provides greater improvements for models that output bounding boxes compared to those output points. This is because when models predict bounding boxes, the bounding boxes inherently reflect the regions that the models are attending to. In contrast, for models predict points, when we manually expand a point into a bounding box, we are simulating the model's attention region. This may fail to accurately represent the actual region that the model focuses on, which further introduces biases in identifying the consensus regions, thus limiting the performance

Table 1: Performance (%) of the proposed test-time scaling method **GUI-RC** across GUI-Grounding benchmarks. Icons refer to output styles: Point (), Bounding-box ().

	SSv2.Mobile		SSv2.Desktop		SSv2.Web		SSv2.avg	SSv1.avg	SSPro.avg
	Text	Icon	Text	Icon	Text	Icon			
General Models									
InternVL3-2B-Instruct	89.92	76.44	38.89	26.19	46.43	25.32	52.75	51.02	1.03
w/ **GUI-RC**	89.92	77.49↑	38.33	24.60	46.07	27.00↑	52.91+0.16	52.20+1.18	1.33+0.30
InternVL3-8B-Instruct	94.19	79.58	79.44	53.17	91.07	71.73	80.97	79.72	13.28
w/ **GUI-RC**	94.19	81.15↑	80.56↑	56.35↑	91.07	71.73	81.68+0.71	80.03+0.31	12.46-0.82
Qwen2.5-VL-3B-Instruct	97.67	75.92	85.56	59.52	84.64	65.82	80.11	76.97	20.18
w/ **GUI-RC**	98.84↑	77.49↑	90.00↑	64.29↑	87.14↑	67.93↑	82.63+2.52	78.46+1.49	23.59+3.41
Qwen2.5-VL-7B-Instruct	98.84	84.29	86.67	73.81	88.57	78.90	86.48	84.20	19.80
w/ **GUI-RC**	99.92↑	85.86↑	91.11↑	73.02	91.79↑	81.43↑	88.52+2.04	85.53+1.33	23.97+4.17
GUI-specific Models									
UGround-V1-7B	96.51	82.72	96.11	82.54	92.50	83.12	89.62	87.11	31.50
w/ **GUI-RC**	96.51	83.77↑	95.56	84.13↑	92.86↑	81.43	89.62+0.00	87.34+0.23	31.63+0.13
UI-TARS-1.5-7B	96.51	86.39	95.00	87.30	88.21	86.50	90.17	87.74	40.92
w/ **GUI-RC**	96.12	86.91↑	96.11↑	90.48↑	90.36↑	86.50	91.12+0.95	88.52+0.78	41.18+0.26
OS-Atlas-Base-7B	91.47	72.25	88.33	64.29	86.43	72.57	80.82	79.80	18.41
w/ **GUI-RC**	91.47	78.53↑	88.89↑	68.25↑	89.29↑	76.37↑	83.57+2.75	81.45+1.65	19.67+0.16

Table 2: Performance (%) of the proposed test-time reinforcement learning method **GUI-RCPO** across GUI-Grounding benchmarks. Icons refer to output styles: Point (), Bounding-box ().

	SSv2.Mobile		SSv2.Desktop		SSv2.Web		SSv2.avg	SSv1.avg	SSPro.avg
	Text	Icon	Text	Icon	Text	Icon			
General Models									
Qwen2.5-VL-3B-Instruct	97.67	75.92	85.56	59.52	84.64	65.82	80.11	76.97	20.18
w/ **GUI-RCPO**	98.06↑	81.68↑	91.11↑	65.08↑	90.71↑	73.42↑	85.14+5.03	82.47+5.50	24.67+4.49
Qwen2.5-VL-7B-Instruct	98.84	84.29	86.67	73.81	88.57	78.90	86.48	84.20	19.80
w/ **GUI-RCPO**	98.84	87.43↑	91.11↑	76.19↑	92.5↑	80.17↑	88.92+2.48	86.64+2.44	25.93+6.13
GUI-specific Models									
UI-TARS-1.5-7B	96.51	86.39	95.00	87.30	88.21	86.50	90.17	87.74	40.92
w/ **GUI-RCPO**	97.29↑	86.39	97.22↑	82.54	91.07↑	87.34↑	90.96+0.79	88.60+0.86	41.43+0.51

of GUI-RC. Nevertheless, GUI-RC still brings improvements to most point-style prediction models, indicating its robustness.

4.2.2 Evaluation Results of GUI-RCPO Experiments

GUI-RCPO is supervised by GUI-RC yet outperforms it. We compare the performance of base models with and without further TTRL training via GUI-RCPO on the three benchmarks. As Table 1 shows, GUI-RCPO also brings consistent improvements and even outperforms GUI-RC. For instance, GUI-RC brings an improvement of 1.49% for Qwen2.5-VL-3B-Instruct on ScreenSpot, while after GUI-RCPO training, it achieves an impressive gain of 5.5%. Intuitively, the performance upper bound of GUI-RCPO should be that of GUI-RC, as it utilizes the region consistency as a reward signal for RL, which seems analogous to directly supervised fine-tuning (SFT) the model with the consensus region. However, in practice, GUI-RCPO not only matches but exceeds GUI-RC, which aligns with prior findings in TTRL (Zuo et al., 2025). This indicates that the model learns a more effective GUI grounding strategy through GUI-RCPO, rather than merely fitting to the consensus region. Notably, GUI-RCPO further brings performance gains for models that have already been specifically trained on GUI tasks, indicating the effectiveness of introducing the region consistency reward. Furthermore, GUI-RCPO provides additional improvements over GUI-RC even

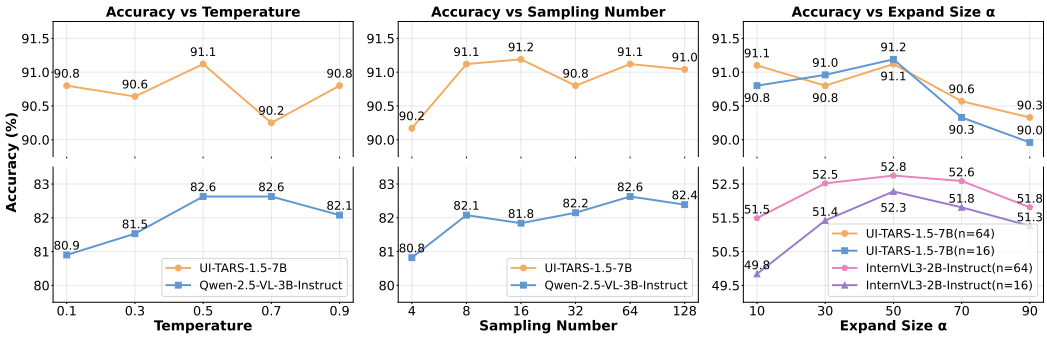


Figure 2: Ablation study results on ScreenSpot-v2 with varying temperature, sampling number, and hyperparameter α.

for point-style prediction models, despite the heightened risk of introducing biases when estimating their attention regions during training, indicating the robustness of this method.

GUI-RCPO generalizes well in out-of-distribution scenarios. Although models are trained on Screenspot-v2, GUI-RCPO also shows significant improvement on ScreenSpot-Pro, which is an out-of-distribution benchmark featuring high-resolution and domain-specific GUIs. This further proves that GUI-RCPO does not rely on overfitting but genuinely enhances the model's general GUI grounding capability. Unlike direct fine-tuning on labeled data, which risks overfitting to the resolution and layout of the training set, GUI-RCPO enables robust generalization across different screen resolutions and interface layouts.

5 Analysis

5.1 Ablation Studies on Decoding Strategy of GUI-RC

We conduct ablation studies on the decoding strategy of GUI-RC to analyze how different parameters affect its GUI grounding performance. Specifically, we first fix the temperature, sampling number, and expand size hyperparameter α to 0.5, 64, and 50 respectively, then vary each parameter individually to observe its impact on GUI-RC. We employ Qwen2.5-VL-3B-Instruct (representing bbox-style prediction models), UI-TARS-1.5-7B and InternVL3-2B-Instruct (representing point-style prediction models) for ablation studies on the ScreenSpot-v2 benchmark. The results are shown in Figure 2.

Temperature. The performance of GUI-RC generally exhibits an increasing-then-decreasing trend as the temperature rises. As temperature controls the diversity of sampled outputs during decoding, a high temperature encourages broad exploration but also increases instability in model generation. Therefore, a moderate increase in temperature helps the model explore broader regions while generating relatively concentrated outputs. However, further increasing the temperature would cause the predicted regions to become overly dispersed, making it difficult to obtain a focused and reliable consensus region.

Sampling Number. As the number of sampled predictions increases, the performance of GUI-RC initially improves and then gradually plateaus. This is because with more samples, the distribution of predicted regions becomes more stable, leading the consensus region to converge toward a fixed area. Once the predictions reach sufficient diversity and coverage, additional samples contribute diminishing improvements, leading to performance saturation.

Hyperparameter α. The performance of GUI-RC follows an increasing-then-decreasing trend as α grows. As the hyperparameter α primarily affects the estimation of the attention area for point-style prediction models, both overly small and large expansion sizes introduce deviations in

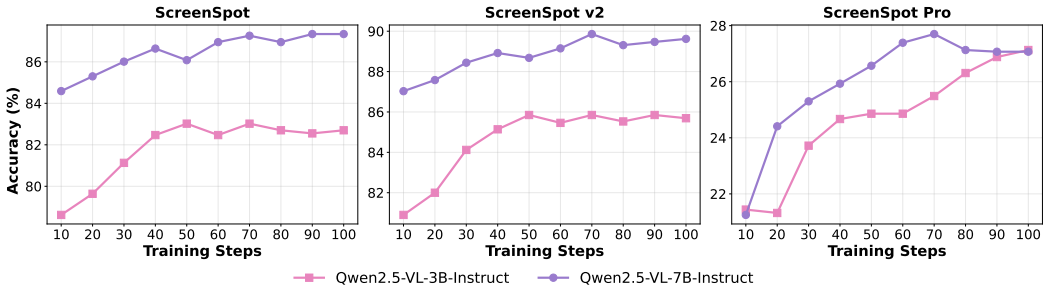


Figure 3: Accuracy (%) across training steps of Qwen2.5-VL-3B-Instruct and Qwen2.5-VL-7B-Instruct throughout GUI-RCPO.

estimating the area models actually attend to. This results in larger deviations in the computed consensus region, thereby impairing the grounding accuracy.

5.2 GUI-RCPO Enables Consistent Improvements during Test-time Training

We observe the performance trajectories of Qwen2.5-VL-3B-Instruct and Qwen2.5-VL-7B-Instruct during GUI-RCPO training on three benchmarks, as shown in Figure 3. As training steps increase, the models' accuracy stably improves across all three GUI-Grounding benchmarks and converges around 80 steps. In particular, although the models are trained solely on ScreenSpot-v2, they do not exhibit overfitting to the training data or degradation on other benchmarks like ScreenSpot-Pro, demonstrating the robust generalization of GUI-RCPO.

5.3 Applying GUI-RC after GUI-RCPO Leads to Further Improvements

We further apply GUI-RC to the bbox-style prediction models after being trained with GUI-RCPO, and evaluate their performance on ScreenSpot-v2 and ScreenSpot-Pro. It is important to note that for GUI-RC in this experiment, we keep all other parameters consistent with previous settings, except increasing the sampling temperature to 1.0. This is because the reward signals in GUI-RCPO training encourage the model to predict more concentrated regions. Therefore, during GUI-RC, a higher decoding temperature is needed to encourage the model to explore broader regions.

Table 3: Performance of applying GUI-RC to bbox-style prediction models after GUI-RCPO on ScreenSpot-v2 and ScreenSpot-Pro benchmarks.

Model	SSv2.avg	SSPro.avg
Qwen2.5-VL-3B-Instruct+GUI-RCPO	85.14	24.67
w/ **GUI-RC**	86.32$_{+1.18}$	26.19$_{+1.52}$
Qwen2.5-VL-7B-Instruct+GUI-RCPO	88.92	25.93
w/ **GUI-RC**	89.78$_{+0.86}$	26.69$_{+0.76}$

The evaluation results are shown in Table 3. It can be observed that even after GUI-RCPO training, applying GUI-RC voting mechanism still leads to additional performance gains. For instance, Qwen2.5-VL-3B-Instruct can gain an additional 1.52% performance on ScreenSpot-Pro through GUI-RC even after GUI-RCPO. This indicates that our methods enable the models to progressively improve themselves in a self-bootstrapping manner, without relying on any external supervision.

6 Discussion

6.1 Why does GUI-RC works?

GUI-RC mitigates two types of hallucinations in GUI grounding during model prediction: misleading and biased (Tao et al., 2025). We show the cases of these two hallucinations in Appendix B.

Mitigate misleading hallucinations. Due to the complexity of GUI layouts and the ambiguity of UI element semantics, models often struggle to accurately align user instructions with the

specific location of the target elements. As a result, models may generate multiple high-confidence prediction targets and struggle to decide on a single region or only be able to predict a large and ambiguous region. GUI-RC addresses these misleading hallucinations by generating multiple predictions and obtaining the consensus region through voting, which enables the model to generate a more confident and spatially precise answer.

Mitigate biased hallucinations. The root cause of grounding biases lies in the granularity mismatch between pixel-level coordinate predictions and patch-level processing of modern vision encoders (Wu et al., 2025b). This mismatch causes models to potentially introduce biases in grounding. GUI-RC aggregates multiple sampled regions and extracts the area model most attends to, thereby resolving the inherent deviations in sampling and reducing grounding biases.

6.2 Limitations

Performance gain limited in point-style grounding. Despite its effectiveness, GUI-RC has several limitations. As analyzed in the experiment section, GUI-RC brings relatively limited improvements for models with point-style outputs. Although most existing GUI Agents adopt point-style grounding in order to seamlessly integrate with subsequent action execution, an increasing number of recent studies (Wu et al., 2025b; Zhou et al., 2025) have pointed out the limitations of point-based prediction and the advantages of region-level supervision. Therefore, we expect that the strengths of GUI-RC will be amplified in future research.

Rely on the model's inherent capabilities. Moreover, GUI-RC primarily addresses misleading and biased hallucinations in grounding, but it is hard to resolve confusion hallucinations (i.e., the predicted region fails to match any valid UI element). In other words, GUI-RC assumes that the model has a certain ability to recognize the target element. It can tolerate the model's predictions to be imprecise or biased, but not completely random or unrelated. Therefore, GUI-RC requires the model to be familiar with the GUI environment, but it does not require the model to be specifically trained on GUI tasks.

7 Conclusion

We introduce GUI-RC, a test-time scaling approach for GUI grounding that leverages region consistency across multiple predictions to enhance model performance without requiring additional training. Building on this idea, we further proposed GUI-RCPO, a test-time reinforcement learning method that transforms region consistency into a self-supervised reward signal, enabling models to self-improve during inference without the need for labeled data. Extensive experiments across a wide range of general and GUI-specific models demonstrate that our methods consistently improve GUI grounding performance and generalize well to out-of-distribution scenarios. Our findings reveal the untapped potential of test-time training for GUI agents and suggest a promising direction toward more robust and data-efficient GUI automation systems.

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents, 2024. URL https://arxiv.org/abs/2401.10935.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training, 2025. URL https://arxiv.org/abs/2501.17161.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents, 2024. URL https://arxiv.org/abs/2410.05243.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking, 2025. URL https://arxiv.org/abs/2501.04519.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use, 2025.

Yexiang Liu, Zekun Li, Zhi Fang, Nan Xu, Ran He, and Tieniu Tan. Rethinking the role of prompting strategies in llm test-time scaling: A perspective of probability theory. *arXiv preprint arXiv:2505.10981*, 2025.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. 2025. URL https://arxiv.org/abs/2503.21620.

Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. Gui-r1 : A generalist r1-style vision-language action model for gui agents. 2025a. URL https://arxiv.org/abs/2504.10458.

Tiange Luo, Lajanugen Logeswaran, Justin Johnson, and Honglak Lee. Visual test-time scaling for gui agent grounding, 2025b. URL https://arxiv.org/abs/2505.00684.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. Ui-tars: Pioneering automated gui interaction with native agents, 2025. URL https://arxiv.org/abs/2501.12326.

ByteDance Seed. Ui-tars-1.5. https://seed-tars.com/1.5, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, Ruochen Xu, and Tiancheng Zhao. Vlm-r1: A stable and generalizable r1-style large vision-language model, 2025. URL https://arxiv.org/abs/2504.07615.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueting Zhuang. Gui-g^2: Gaussian reward modeling for gui grounding, 2025a. URL https://arxiv.org/abs/2507.15846.

Fei Tang, Yongliang Shen, Hang Zhang, Siqi Chen, Guiyang Hou, Wenqi Zhang, Wenqiao Zhang, Kaitao Song, Weiming Lu, and Yueting Zhuang. Think twice, click once: Enhancing gui grounding via fast and slow systems, 2025b. URL https://arxiv.org/abs/2503.06470.

Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, Kaitao Song, Jian Shao, Weiming Lu, Jun Xiao, and Yueting Zhuang. A survey on (m)llm-based gui agents, 2025c. URL https://arxiv.org/abs/2504.13865.

Xingjian Tao, Yiwei Wang, Yujun Cai, Zhicheng Yang, and Jing Tang. Understanding gui agent localization biases through logit sharpness. *arXiv preprint arXiv:2506.15425*, 2025.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception, 2024. URL https://arxiv.org/abs/2401.16158.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Hang Wu, Hongkai Chen, Yujun Cai, Chang Liu, Qingwen Ye, Ming-Hsuan Yang, and Yiwei Wang. Dimo-gui: Advancing test-time scaling in gui grounding via modality-aware visual reasoning, 2025a. URL https://arxiv.org/abs/2507.00008.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. *arXiv preprint arXiv:2506.03143*, 2025b.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. Os-atlas: A foundation action model for generalist gui agents, 2024. URL https://arxiv.org/abs/2410.23218.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction, 2024. URL https://arxiv.org/abs/2412.04454.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Yuqi Zhou, Sunhao Dai, Shuai Wang, Kaiwen Zhou, Qinglin Jia, and Jun Xu. Gui-g1: Understanding r1-zero-like training for visual grounding in gui agents. 2025. URL https://arxiv.org/abs/2505.15810.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models, 2025. URL https://arxiv.org/abs/2504.10479.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning, 2025. URL https://arxiv.org/abs/2504.16084.

A Pseudo-code of Region Consistency Reward Function

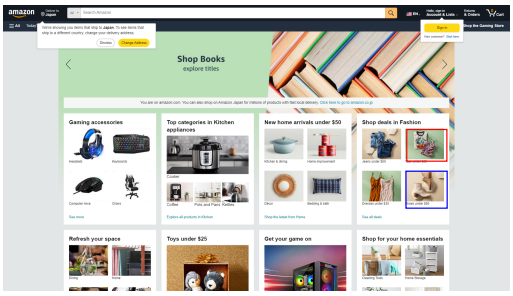
```python
def region_consistency_reward_fn(outputs, image_size):
    """
    Assigns a soft reward to each output based on how much the predicted bounding box consists
      with the most voted region.
    """
    # Initialize voting grid
    W, H = image_size
    grid = zeros(H, W)
    sampled_bboxes = []

    # Vote on each bounding box
    for output in outputs:
        bbox = extract_answer(output["content"])
        x1, y1, x2, y2 = int(bbox[0]), int(bbox[1]), int(bbox[2]), int(bbox[3])
        sampled_bboxes.append([x1, y1, x2, y2])
        grid[y1:y2, x1:x2] += 1

    # Find the most voted region
    max_vote = max(grid)

    # Compute reward for each bounding box
    rewards = []
    for bbox in sampled_bboxes:
        x1, y1, x2, y2 = bbox
        region = grid[y1:y2, x1:x2]
        region_sum = sum(region)
        region_area = (x2 - x1) * (y2 - y1)
        if region_area == 0:
            reward = 0
        else:
            reward = region_sum / (max_vote * region_area)
        rewards.append(reward)

    return rewards

def extract_answer(content, alpha):
    """
    Extracts bounding box coordinates from a text string.
    If 4 numbers are found, treat as a full box [x1, y1, x2, y2].
    If 2 numbers are found, treat as a single point [x, y] and convert to an alpha*alpha box.
    Otherwise, return a zero box.
    """
    numbers = find_all_numbers(content)

    if length(numbers) == 4:
        return [float(numbers[0]), float(numbers[1]),
                float(numbers[2]), float(numbers[3])]

    elif length(numbers) == 2:
        x, y = float(numbers[0]), float(numbers[1])
        return [x - alpha/2, y - alpha/2, x + alpha/2, y + alpha/2]

    else:
        return [0, 0, 0, 0]
```

Listing 1: The pseudo-code of the region consistency reward function.

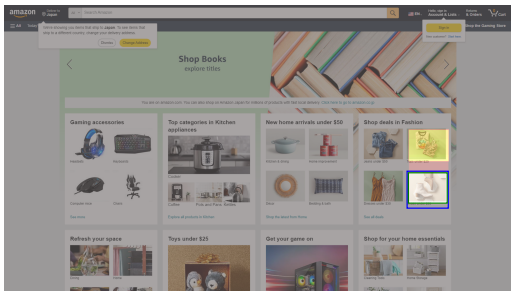
B Case Studies on How GUI-RC Mitigates Hallucinations in GUI Grounding

To provide a more intuitive understanding on how GUI-RC mitigates two types of hallucinations, **misleading** and **biased**, during GUI grounding, we present typical failure cases drawn from the evaluation results of Qwen2.5-VL-7B-Instruct, and show how the model successfully localizes the target elements after applying GUI-RC. These examples are illustrated in Figure 4.

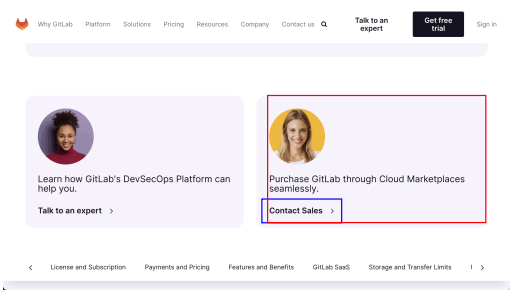
Mitigating misleading hallucinations. Due to the complexity of GUI layouts and the semantic similarity between UI elements, models often get confused when aligning instructions with potential target elements. As shown in Figure 4a, the instruction asks *"check shoes under 50 dollars in 'shop deals in fashion' part"*, but under greedy decoding, the model mistakenly selects the region of *"tops under 25 dollars"*. This may be caused by a misunderstanding of the elements' semantics, leading to



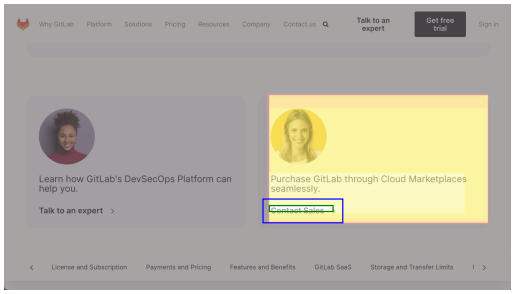
(a) ScreenSpot-v2: "check shoes under 50 dollars in 'shop deals in fashion' part" (Greedy)



(b) ScreenSpot-v2: "check shoes under 50 dollars in 'shop deals in fashion' part" (GUI-RC)



(c) ScreenSpot-v2: "contact sales" (Greedy)



(d) ScreenSpot-v2: "contact sales" (GUI-RC)

Figure 4: Case studies of how GUI-RC mitigates two types of hallucinations in GUI grounding. In each row, the left image shows the **Greedy Decoding** result, where the blue box denotes the ground truth, and the red box denotes the model's prediction. The right image shows the spatial voting heatmap obtained after applying **GUI-RC**. The brighter regions reflect higher region consistency, and the green box denotes the extracted **consensus region**.

an incorrect match with the intended target. After applying GUI-RC, the model performs multiple samplings and constructs a spatial voting grid to identify the region with the highest prediction consistency across samples. As shown in Figure 4b, this consensus region successfully matches the ground-truth bounding box, and effectively mitigates the misleading hallucinations.

Mitigating biased hallucinations. Biased hallucinations primarily stem from the granularity mismatch between pixel-level coordinate predictions and patch-level representations processed by modern visual encoders. As shown in Figure 4c, the instruction asks *"contact sales"*. Although the model understands the general target location, its direct prediction encompasses the entire contact card rather than precisely identifying the *"Contact Sales"* button, resulting in failed grounding. GUI-RC mitigates this issue by aggregating the regions from multiple predictions and extracting the most attention-concentrated area during sampling. As shown in Figure 4d, the consensus region precisely matches the location of the target element, enabling successful grounding and significantly improving the model's robustness.

C Evaluation Details

This section provides an overview of the benchmarks, models, and prompt templates used in the evaluation of GUI grounding performance.

GUI Grounding Benchmarks

- **ScreenSpot** (Cheng et al., 2024) is a widely used benchmark for GUI grounding, containing over 600 multi-platform screenshots with 1,272 instructions across mobile, web, and desktop domains. Each sample consists of a natural language instruction, a screenshot

and the corresponding target region annotated as a bounding box. It serves as a standard zero-shot evaluation dataset.

- **ScreenSpot-v2** (Wu et al., 2024) is an enhanced version of ScreenSpot with improved annotation quality. It contains 1,272 instructions with balanced distribution across platforms. Annotation errors from the original dataset (e.g., ambiguous or incorrect labels) were manually corrected to ensure more reliable benchmarking.
- **ScreenSpot-Pro** (Li et al., 2025) focuses on GUI grounding in high-resolution professional interfaces, covering 23 expert software applications across five professional categories and three operating systems. The dataset challenges models with complex UI layouts and fine-grained target regions. It is designed to evaluate generalization to real-world professional software.

Model Details

- **Qwen2.5-VL** (Bai et al., 2025) is a multimodal model from Alibaba based on the Qwen2.5 language model, extended with a visual encoder trained from scratch. It supports fine-grained localization via point or bounding box output, and is optimized for multiple multimodal tasks.
- **InternVL3** (Zhu et al., 2025) is an open-source vision-language model trained jointly on vision and text modalities. Unlike adapter-based approaches, it is trained from scratch with a unified architecture and achieves strong performance on multimodal tasks.
- **UGround** (Gou et al., 2024) is a GUI-specific grounding model based on LLaVA. It is trained on a large-scale synthetic dataset of GUI screenshots and instructions, enabling pixel-level grounding from vision alone, without relying on DOM or accessibility metadata.
- **OS-Atlas** (Wu et al., 2024) is a foundation model for GUI action grounding, trained on millions of synthetic screenshots spanning five platforms. It uses unified action formats and multi-stage instruction tuning to produce robust GUI agents.
- **UI-TARS** (Qin et al., 2025) is an end-to-end GUI agent that directly outputs interaction actions from screenshots and instructions. It unifies perception, grounding, and multi-step planning in a single model and is trained via interaction feedback collected from both simulated environments and real-world environments.

Prompts We list the prompt templates used for each model in Table 4. For models that provide official prompt templates for GUI grounding, such as InternVL3, UGround, and OS-Atlas, we adopt the same prompts as specified in their original implementations to ensure fair comparison.

Table 4: Prompt templates used for different models in GUI grounding evaluation. The {instruction} placeholder is replaced with the actual input at inference time.

Model	Prompt Template
Qwen2.5-VL	You are a GUI analysis expert. Based on the screenshot, your task is to locate the UI element that best matches the instruction: '{instruction}'. You MUST output exactly one bounding box in the format [x1, y1, x2, y2] strictly. Do not include any explanations, descriptions, or additional text. **Output format: [x1, y1, x2, y2]**
InternVL3	SYSTEM: You are InternVL, a GUI agent. You are given a task and a screenshot of the screen. You need to perform a series of pyautogui actions to complete the task. <image> USER: '{instruction}'.
UGround	Your task is to help the user identify the precise coordinates (x, y) of a specific area/element/object on the screen based on a description. - Your response should aim to point to the center or a representative point within the described area/element/object as accurately as possible. - If the description is unclear or ambiguous, infer the most relevant area or element based on its likely context or purpose. - Your answer should be a single string (x, y) corresponding to the point of the interest. Description: '{instruction}'.
OS-Atlas	In this UI screenshot, what is the position of the element corresponding to the command '{instruction}' (with bbox)?
UI-TARS	You are a GUI analysis expert. Based on the screenshot, your task is to identify the center point of the UI element that best matches the instruction: '{instruction}'. You MUST output exactly one point coordinate in the format [x, y] strictly. Do not include any explanations, descriptions, or additional text. **Output format: [x, y]**